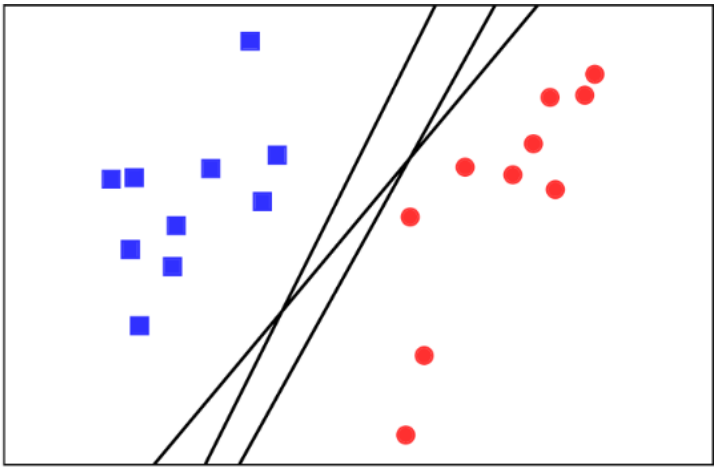
Support Vector Machine

Hoàng Vũ Minh

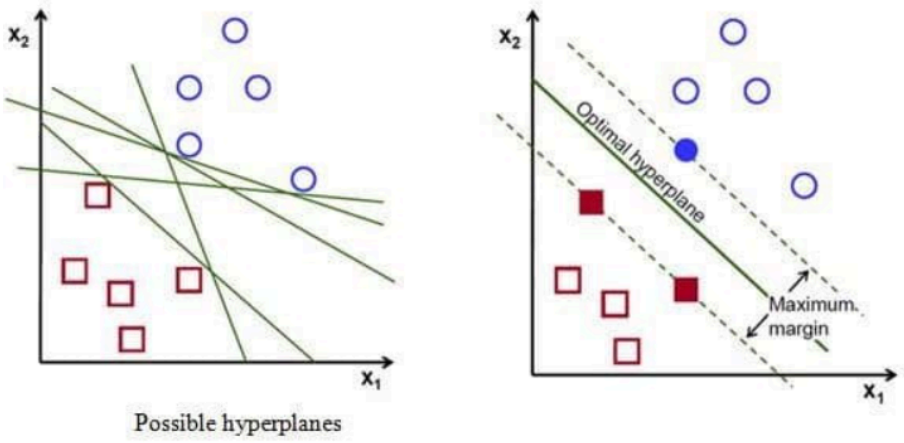
Support Vector Machines (có tài liệu dịch là Máy véctơ hỗ trợ) là một trong số những thuật toán phổ biến và được sử dụng nhiều nhất trong học máy trước khi mạng nơ ron nhân tạo trở lại với các mô hình deep learning, nó cũng là một trong những thuật toán classification nổi tiếng và được sử dụng nhiều nhất trong ML bên cạnh Softmax Regression. Phần toán học của SVM rất phức tạp nên trong báo cáo này, em sẽ trình bày chủ yếu hơn về mặt ý tưởng và những điều (cần thiết / hay) trong SVM.

1. Ý tưởng cơ bản:

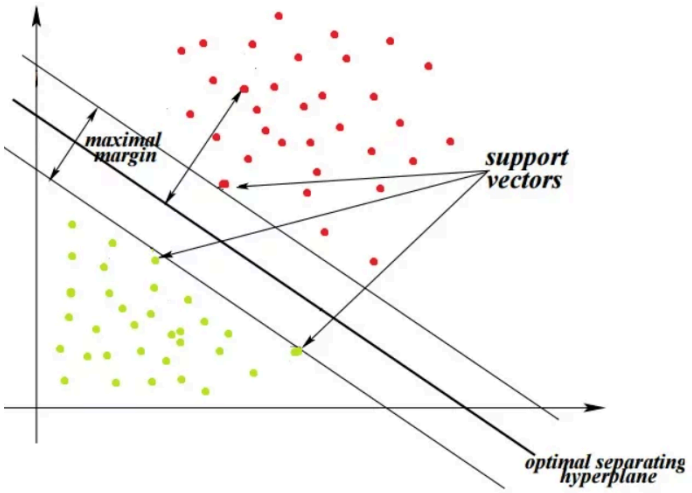
Ý tưởng cơ bản của SVM vẫn là tìm ra một đường thẳng / siêu phẳng để phân cách các tập điểm ra theo một cách nào đó. Nếu dữ liệu là linear separable (có thể phân cách bằng đường thẳng / mặt phẳng hoặc tuyến tính tương tự) thì ta biết sẽ có vô số đường thẳng / mặt phẳng có thể làm điều đó.



Vì vậy nên trong SVM tồn tại thêm 1 khái niệm là lề (margin), qua đó giới hạn số mặt phẳng lại, hay thật ra là để chọn ra mặt phẳng duy nhất, là tốt nhất để phân chia 2 tập điểm này. Tốt nhất ở đây là có lề rộng nhất, lề là một khoảng sao cho cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau, như vậy về mặt ý tưởng, ***SVM đi tìm đường phân cách có lề rộng nhất***.

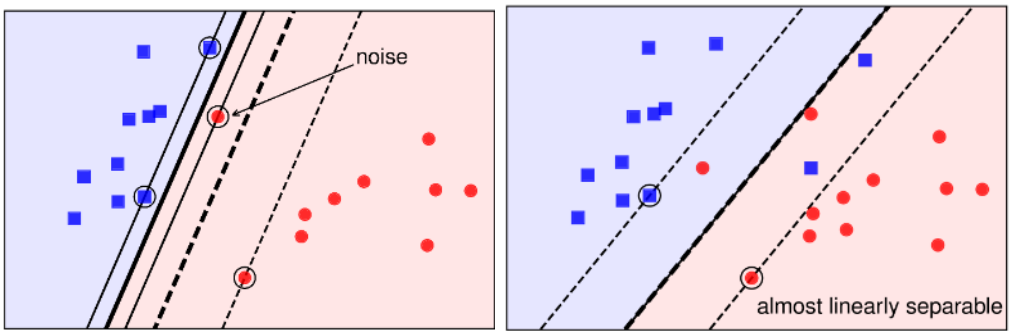


Các điểm dữ liệu nằm trên hoặc gần nhất với siêu phẳng được gọi là véc tơ hỗ trợ, chúng ảnh hưởng đến vị trí và hướng của siêu phẳng. Các véc tơ này được sử dụng để tối ưu hóa lề và nếu xóa các điểm này, vị trí của siêu phẳng sẽ thay đổi.



2. Hard Margin và Soft Margin SVM

Dựa vào khái niệm về lề, ta có thể chia SVM thành 2 loại: Hard Margin SVM và Soft Margin SVM. Hard Margin là ý tưởng như đã trình bày ở trên, tìm một đường phân cách 2 class mà khoảng cách từ điểm gần nhất giữa mỗi class đến đường phân cách là như nhau, gọi là hard margin vì khi đó lề được fix cố định. Tuy nhiên trên thực tế, nhiều khi dữ liệu có thể có các noise…. dẫn đến việc nếu kiên quyết sử dụng lề cứng, kích thước lề sẽ bị nhỏ đi, trong khi mục tiêu của chúng ta vẫn là ***tìm đường phân cách sao cho lề lớn nhất có thể***.



Các hình trên là 2 trường hợp xấu với hard margin, như trong hình 1, 1 điểm noise đỏ làm lề trở nên rất nhỏ, hình 2 là kiểu hai lớp dữ liệu này gần như linaer separable (chứ không phân chia được, nó chỉ là gần như), nên nếu làm Hard Margin thì coi như bài toán vô nghiệm. Trong khi đó nếu hi sinh một vài điểm noise, ta có thể tìm được lề tốt hơn rất nhiều.

-> Ý tưởng của Soft Margin: ***Hi sinh một vài điểm noise để tìm được lề tốt hơn, rộng hơn.***

3. Bài toán tối ưu:

Bài toán tối ưu trong Support Vector Machine (SVM) chính là bài toán đi tìm đường phân chia sao cho margin là lớn nhất.

3.1: Khoảng cách từ một vector đến một siêu phẳng:

Khoảng cách từ một điểm x_0 đến siêu phẳng $wTx + b = 0$ được tính bằng công thức:

$$\frac{|\mathbf{w}^T\mathbf{x}_0 + b|}{||\mathbf{w}||_2}$$

Trong đó: $||\mathbf{w}||_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$ với d là số chiều

3.2: Bài toán tối ưu cho Hard Margin

Với Hard Margin, bài toán tối ưu là tìm lề lớn nhất:

$$\text{margin} = \min_n \frac{y_n(\mathbf{w}^T\mathbf{x}_n + b)}{||\mathbf{w}||_2}$$

Nhìn chung ta cần tìm w, b sao cho

$$(\mathbf{w}, b) = \arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2$$
$$\text{subject to:} \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N$$

3.3: Bài toán tối ưu cho Soft Margin

$$(\mathbf{w}, b, \xi) = \arg\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{n=1}^{N}\xi_n$$
$$\text{subject to:} \quad 1 - \xi_n - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \ldots, N$$
$$-\xi_n \leq 0, \ \forall n = 1, 2, \ldots, N$$

Trong đó:

- C là một số dương có thể do tự đặt hoặc thử nghiệm xem trường hợp nào tốt nhất bằng cross validation. Đây là một siêu tham số quan trọng khi làm việc với SVM vì nó ảnh hưởng trực tiếp đến kết quả.

- Siêu tham số C:
  - Dựa vào công thức ta có thể thấy sự thay đổi của C ảnh hưởng khá lớn đến giá trị của hàm mục tiêu này.
  - Nếu C nhỏ, sự hy sinh cao hay thấp không gây ảnh hưởng nhiều tới giá trị của hàm mục tiêu.
  - Tuy nhiên nếu C rất lớn, để tối ưu hóa hàm mục tiêu, phần về sau sẽ được chú ý đến. Bởi vì C rất lớn, để hàm này nhỏ nhất, phần $\sum_{n=1}^{N}\xi_n$ sẽ tiến dần về 0, hoặc thậm chí bằng 0 luôn.

4. Hinge loss

$$\text{Loss}(y_n, \mathbf{x}_n) = \max(0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

Giải thích một chút: y là nhãn, f(x) là dự đoán.

- Nếu cụm y * f(x) <1, tức là tồn tại sai số, khi đó có nghĩa là cái điểm ta xét nằm trong lề, khi đó sẽ tồn tại điểm phạt.

Nếu trong soft margin mà thêm regurlazation:

$$\text{Loss} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i \cdot f(x_i))$$

5. Thủ thuật Kernel

Ở bên trên, khi ta gặp trường hợp dữ liệu gần như linear separable, soft margin vẫn có thể hoạt động hiệu quả. Tuy nhiên trong những trường hợp phi tuyến hẳn, người ta thường sử dụng Kernel Trick để giúp SVM có thể làm việc với những dữ liệu phi tuyến tính. Kernel đơn giản có thể hiểu là một hàm ánh xạ dữ liệu, thay đổi dữ liệu hiện tại sang không gian khác (nhiều chiều hơn) mà ở đó ta có thể tìm được mặt phẳng phân cách.

Tên	Công thức	`kernel`	Thiết lập hệ số
linear	$\mathbf{x}^T\mathbf{z}$	`'linear'`	không có hệ số
polynomial	$(r + \gamma\mathbf{x}^T\mathbf{z})^d$	`'poly'`	d: `degree`, γ: `gamma`, r: `coef0`
sigmoid	$\tanh(\gamma\mathbf{x}^T\mathbf{z} + r)$	`'sigmoid'`	γ: `gamma`, r: `coef0`
rbf	$\exp(-\gamma\|\|\mathbf{x} - \mathbf{z}\|\|_2^2)$	`'rbf'`	$\gamma > 0$: `gamma`

Dưới đây là 4 kernel thông dụng, việc sử dụng kernel nào có thể được tìm thấy bằng cách sử dụng lưới kernel, hoặc kiểm chứng chéo.

6. SVR (Support Vector Regression)

Cũng dùng ý tưởng của SVM nhưng mà cho hồi quy. Mục tiêu của chúng ta là tìm siêu phẳng (hyperplane) sao cho chứa được tối đa các quan sát trong tập huấn luyện nằm trong khoảng lề epsilon.

- slack variables: Mấy cái biến nằm ngoài lề, như trong hình là mấy chỗ khoanh vàng ngoài lề, thật ra nó k phải biến mà nó kiểu khoảng cách giữa mấy cái sai với lề.

Bài toán tối ưu: như trong hình. C vẫn là điểm phạt, và nếu C lớn thì cái slack ở sau phải cực nhỏ đi.

- Nói kỹ hơn về cái C này một chút: Nó sẽ liên quan đến trade-off về bias và varriance cho mô hình của ta:
    - C to: bias thấp (vì nó phạt mấy điểm sai thấm quá, nên model sẽ cố fit), nhưng varriance lại cao (vì nó cố fit cả noise). Nói chung dễ dẫn đến overfit.